1. Introduction

1.1 Purpose

This document was created by:

- Ashish Ghaskata
- Krishna Raj Bhandari
- Suraj Bhatta
- Mohammad Adnan Khan

It serves as the architecture documentation for the **FunFlip Educational Game**, a mobile memory game targeting children aged 4–6.
This document should be read by:

- Developers and system architects
- UI/UX designers
- Quality assurance and testing teams
- Future maintainers

It is binding for anyone involved in designing, maintaining, or extending the FunFlip application.

1.2 Summary

This documentation outlines the architectural structure of the FunFlip Game.
It covers:

- Functional and non-functional requirements
- System decomposition
- Interface design
- Design decisions and rationale
- Reuse of components
- Human-machine interaction modeling

Stakeholders: development team, testers, product owner, designers.

1.3 Definitions and Abbreviations

Term	**Definition**
FR	Functional Requirement
NFR	Non-Functional Requirement
UX	User Experience
Q	Quality Attribute
L	Legal Requirement
T	Technical Design Element
HMI	Human-Machine Interface

1.4 References, Standards, and Rules

- FunFlip Requirements Specification
- Godot Engine 4.x documentation
- JSON structure used for categories
- Usability guidelines for early childhood apps
- ISO/IEC 25010 quality standards

1.5 Overview

This document is structured as follows:

- Description of core architecture and its principles
- Interfaces and component responsibilities
- Design decisions and alternatives considered
- Component reuse strategy
- Human-machine interface requirements and modeling
- Summary, appendix (diagrams), and index

2. System Architecture

2.1 Functional and Non-Functional Requirements

Req.ID	Requirements	Must/Can	Category
FR_01,UX_01	Allow category selection	Must	FR, UX
FR_02	Select 3 Different Difficulty Level	Must	FR
FR_03	Flip cards and match pairs	Must	FR
FR_04, UX_02, Q_01	Audio on card flip	Must	FR, UX, Q
FR_05	Unmatched cards flip back after delay	Must	FR
FR_06	Display turn count	Must	FR
FR_07,UX_03	Happy sound on match	Must	FR, UX
FR_08, UX_04	Replay, return to menu, quit	Must	FR, UX
FR_09, Q_02	Voice feedback by category	Can	FR, Q
FR_10, UX_01	Toggle sound/music	Can	FR, UX
NFR_01, UX_04, Q_03	Simple, colorful UI for kids	Must	NFR, UX, Q
NFR_02, L_01	Offline operation	Must	NFR, L
NFR_03, Q_01	Quick Start	Must	NFR, Q
NFR_04, L_01	No personal data collection	Must	NFR, L
NFR_05, T_01	Modular Godot components	Can	T, NFR

List of Requirements

Attribute	Definition	How Achieved	Measured By
Usability	Child-friendly and easy to navigate	Large buttons, 3-click access, no text	Game start in ≤ 3 steps
Accessibility	Audio feedback and inclusive design	Voice cues, mute toggle	100% cards play sounds
Performance	Fast and stable response	Optimized scenes and animations	Minimum loading time, no crashes

2.2 Prioritization of Non-Functional Requirements

2.3 Architectural Principles

1. **Strict Layered Architecture** – UI → Scene Loader → Game Logic → Data → Services
2. **Separation of Concerns** – Each module has a clear role
3. **Low Coupling & High Cohesion** – Components interact via clean signals/APIs
4. **Open/Closed Principle** – Easy to add cards or features without rewriting logic
5. **KISS (Keep It Simple)** – Single Card.tscn reused, minimal dependencies
6. **Centralized Cross-Cutting Concerns** – Audio, performance, and usability handled uniformly

3. System Interfaces

SceneLoader

- Handles transitions between all major game scenes
- Exposes:
    - Show start screen
    - Show category/level/game screens
    - Show completion screen

GameManager

- Manages gameplay logic
- Exposes:
    - Start game
    - Handle card flip
    - Pause/Resume
    - Get score
    - End game

DataManager

- Provides and persists card/category data
- Exposes:
    - Get categories/cards
    - Save/load game progress

AudioPlayer

- Manages sound effects and settings
- Exposes:
    - Play sound by name

  - Mute/unmute
  - Set volume

UICallbacks

- Reacts to UI input events
- Exposes:
  - Play button pressed
  - Category/level selection
  - Back navigation

4. System Design

4.1 System Decomposition

- **Presentation Layer**: Start, Category, Level, Game, Completion screens
- **App Controller**: SceneLoader (Main.gd) handles flow
- **Game Logic**: GameManager, Card.gd, ScoreManager, MatchChecker
- **Data Layer**: DataManager loads from categories.json
- **System Services**: AudioControl.gd, Godot's FileAccess API

4.2 Design Decisions

- Use 5-layered architecture for modularity
- Use JSON for extensible card storage
- Use signals for UI → Logic communication
- Centralized scene control with SceneLoader
- Reuse single Card.tscn scene across all levels
- Target smooth 60 FPS
- Record decisions using ADRs (Architecture Decision Records)

4.3 Design Alternatives Considered

Alternative	**Rejected Because**
Embed card data in logic	Violates open/closed principle
Let GameManager control scenes	Breaks separation of concerns
Load card data per level	Slower, reduces responsiveness
Use multiple Card scenes	Increases maintenance complexity
Allow UI to access DataManager	Breaks layering, tight coupling

4.4 Reuse of Components

- **Card.tscn**
    - Used for all cards
    - Reduces duplication, updates propagate globally
- **AudioControl.gd**
    - Central manager for all game audio
    - Ensures consistent volume/logic
- **SceneLoader (Main.gd)**
    - Reused for all transitions
    - Decouples navigation from game logic
- **categories.json**
    - Contains all card data
    - Easy to extend without code changes

5. Human-Machine Interface (HMI)

5.1 Requirements

- Simple, age-appropriate design
- Max 3 steps to gameplay
- Large, touch-friendly buttons
- Visual/audio feedback
- High contrast, toggleable audio

5.2 Design Principles

- **KISS** – No clutter, only essentials shown
- **Consistency** – Uniform button styles
- **Child-friendly** – Big fonts, cheerful colors

- **Accessible** – Audio + visual feedback everywhere

5.3 Interaction Model

- Linear flow:
  Start → Category → Level → Game → Completion
- Input method: Touch
- Feedback loop:
  Tap → card flips
  Match → visual + audio
  Win → completion screen

Scene transitions are all handled by SceneLoader.

6. Summary

The FunFlip Educational Game architecture follows best practices for modularity, performance, usability, and accessibility. It ensures scalability through clean layering, reuse, and design separation.

7. Appendix

- Figures:
  - Figure 1: High-Level System Architecture
  - Figure 2: Activity Diagram
  - Figure 3: Domain Data Model
  - Figure 4: Interaction Modeling

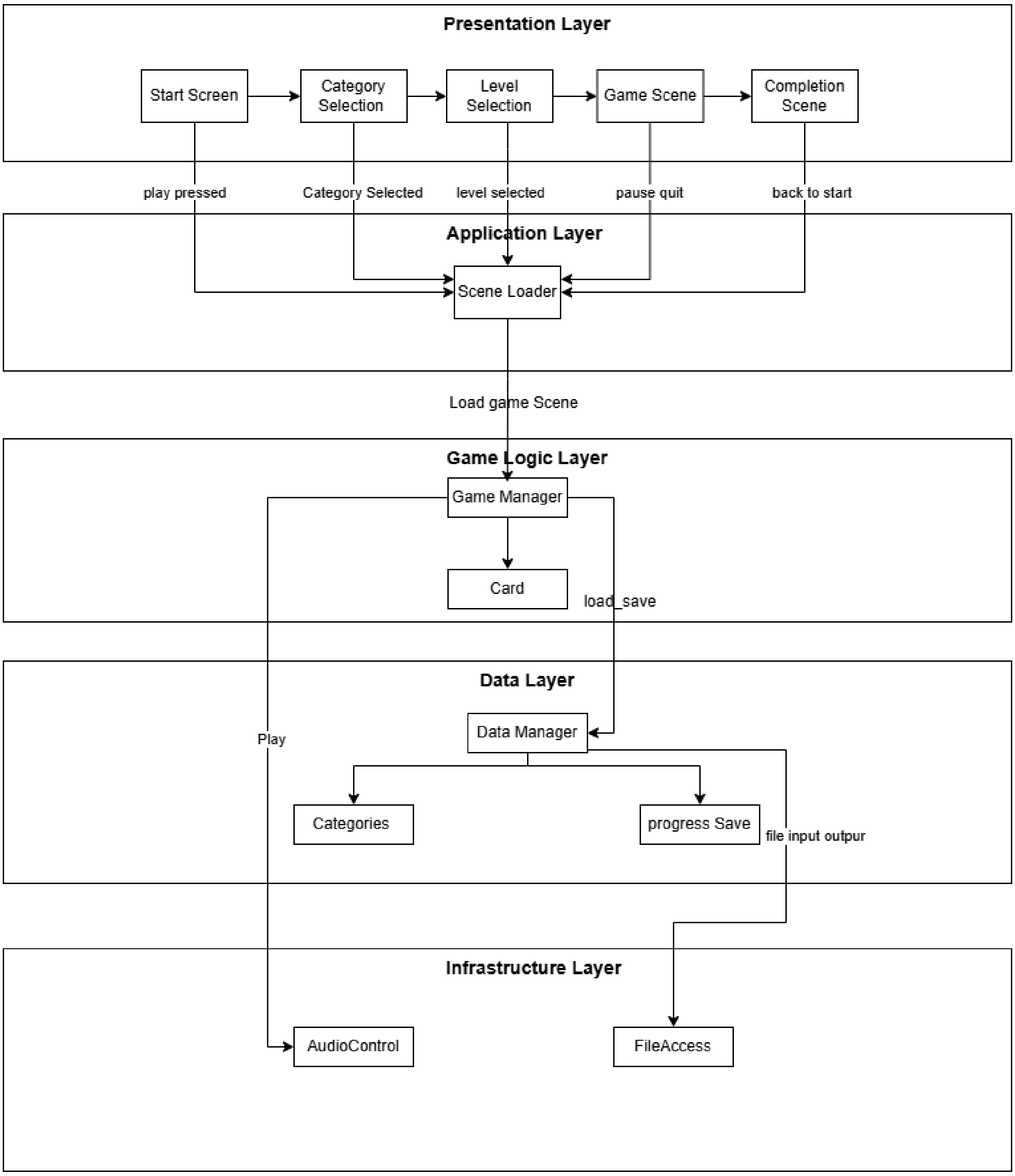

Figure 1 High-Level System Architecture

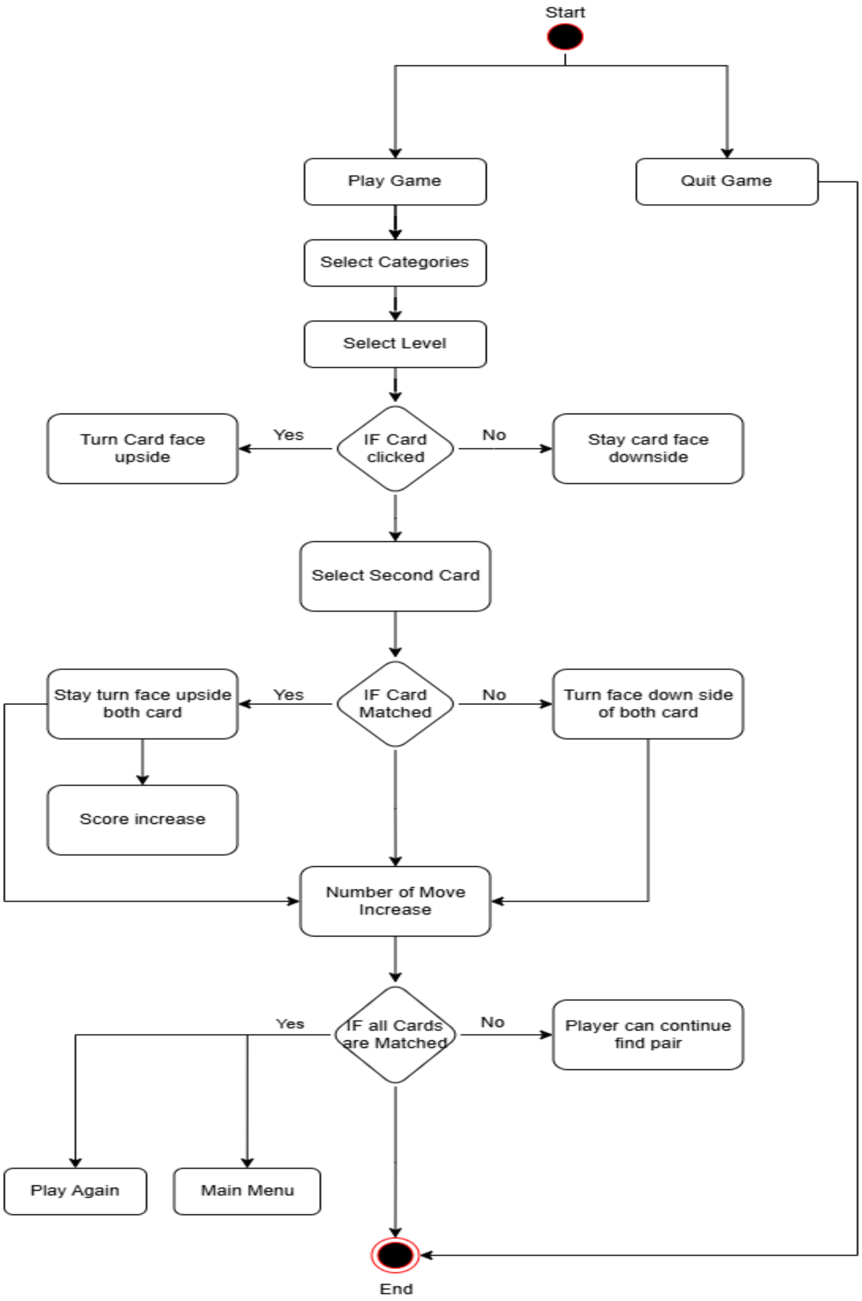

Figure 2 Activity Diagram

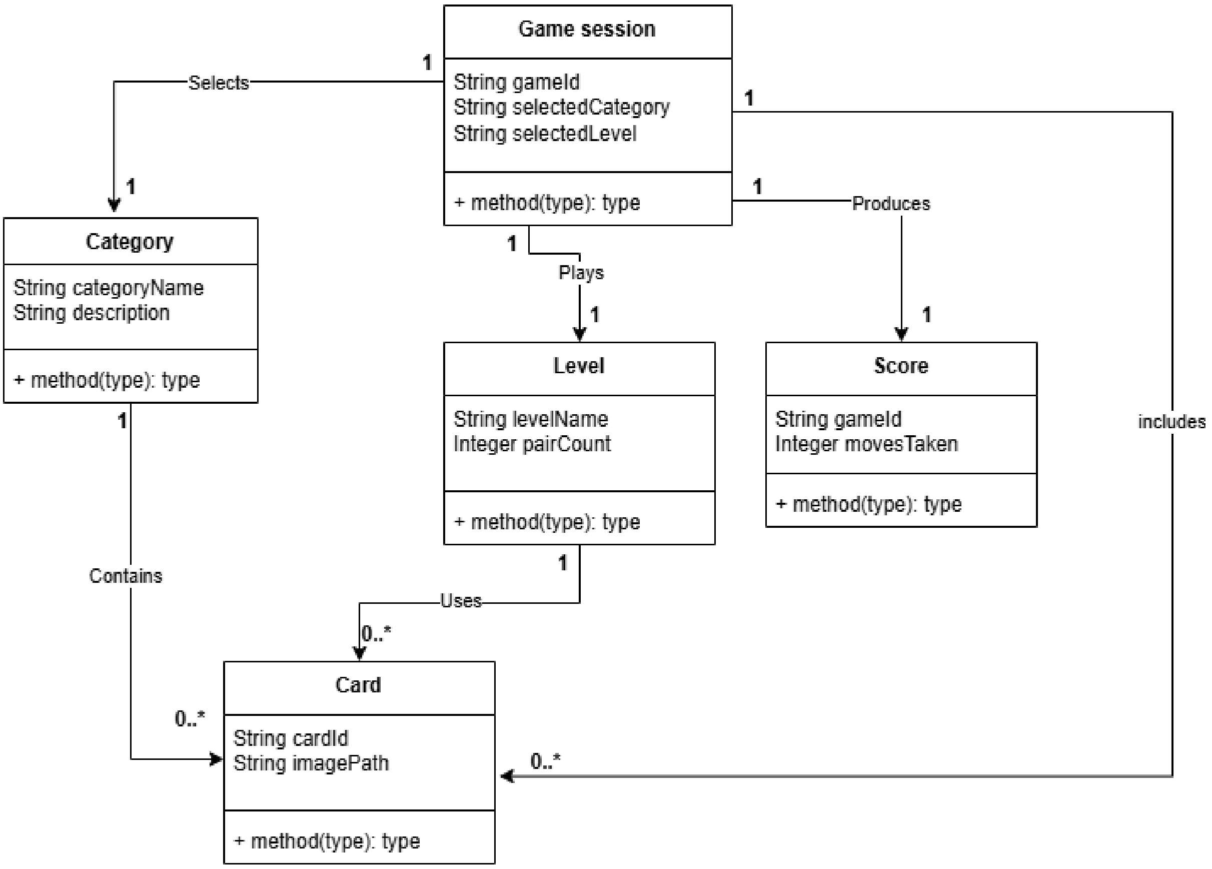

Figure 3 Domain Data Model

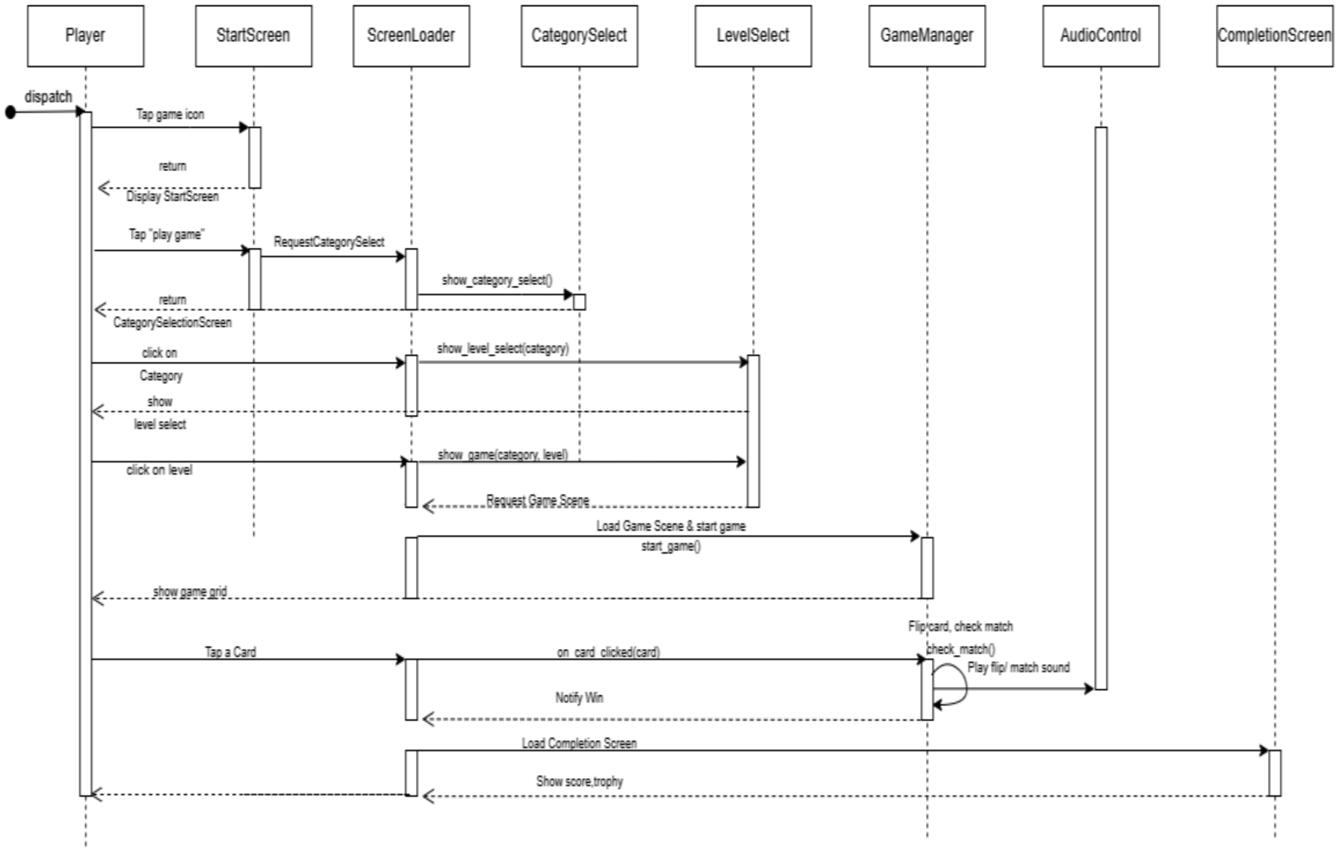

Figure 4 Interaction Modeling

8. Index

Table of Contents